Jurnal Teknologi

Full Paper

A WEAKLY HARD SCHEDULING APPROACH OF PARTITIONED SCHEDULING ON MULTIPROCESSOR SYSTEMS

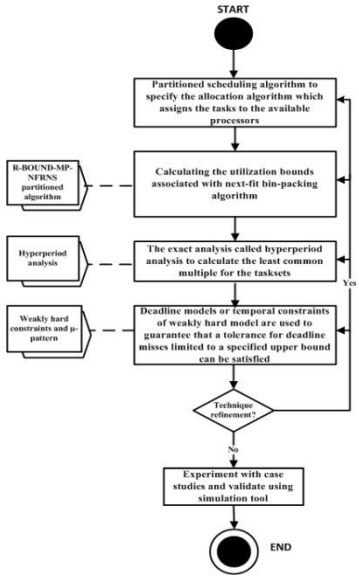
Article history
Received
2 February 2015
Received in revised form
14 September 2015
Accepted
12 October 2015

Habibah Ismail*, Dayang N. A. Jawawi

Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia

*Corresponding author
habibahisma@gmail.com

Graphical abstract



Abstract

Real-time systems or tasks can be classified into three categories, based on the "seriousness" of deadline misses – hard, soft and weakly hard real-time tasks. The consequences of a deadline miss of a hard real-time task can be prohibitively expensive because all the tasks must meet their deadlines whereas soft real-time tasks tolerate "some" deadline misses. Meanwhile, in a weakly hard real-time task, the distribution of its met and missed deadlines is stated and specified precisely. As real-time application systems increasingly come to be implemented upon multiprocessor environments, thus, this study applies multiprocessor scheduling approach for verification of weakly hard real-time tasks and to guaranteeing the timing requirements of the tasks. In fact, within the multiprocessor, the task allocation problem seem even harder than in uniprocessor case; thus, in order to cater that problem, the sufficient and efficient scheduling algorithm supported by accurate schedulability analysis technique is present to provide weakly hard real-time guarantees. In this paper, a weakly hard scheduling approach has been proposed and schedulability analysis of proposed approach consists of the partitioned multiprocessor scheduling techniques with solutions for the bin-packing problem, called R-BOUND-MP-NFRNS (R-BOUND-MP with next-fit-ring noscaling) combining with the exact analysis, named hyperperiod analysis and deadline models; weakly hard constraints and μ-pattern under static priority scheduling. Then, Matlab simulation tool is used in order to validate the result of analysis. From the evaluation results, it can be proven that the proposed approach outperforms the existing approaches in terms of satisfaction of the tasks deadlines.

Keywords: Weakly hard real-time tasks, partitioned scheduling, multiprocessor systems, hyperperiod analysis, deadlines models

Abstrak

Sistem atau tugas masa nyata boleh diklasifikasikan kepada tiga kategori, berdasarkan "kesungguhan" terlepas tarikh akhir – tugas masa nyata yang keras, lembut dan keras yang lemah. Kebarangkalian tugas masa nyata yang keras terlepas tarikh akhir boleh menjadi terlampau mahal kerana semua tugas-tugas mesti memenuhi tarikh akhir mereka sedangkan masa nyata yang lembut boleh bertolak ansur "beberapa" tarikh akhir yang terlepas. Sementara itu, dalam tugas masa nyata keras yang lemah, pengagihan tarikh akhir yang bertemu dan terlepas dinyatakan dengan tepat. Disebabkan peningkatan terhadap aplikasi sistem masa nyata yang dilaksanakan kepada persekitaran multi pemproses, dengan itu, kajian ini mengaplikasikan pendekatan penjadualan multi pemproses untuk pengesahan tugas masa nyata keras yang lemah dan untuk menjamin keperluan masa tugas-tugas. Malah, dalam multi pemproses, masalah peruntukan tugas kelihatan lebih sukar daripada dalam kes uni pemproses; dengan itu, untuk menampung masalah itu, algoritma penjadualan yang cekap disokong oleh teknik penjadualan analisis yang tepat diperkenalkan untuk memberikan jaminan masa nyata keras yang lemah. Dalam penulisan ini, pendekatan penjadualan keras yang lemah telah dicadangkan dan pendekatan penjadualan analisis yang dicadangkan terdiri daripada teknik-teknik

penjadualan multi pemproses dibahagikan dengan penyelesaian untuk masalah pembungkusan-pemproses, yang dikenali sebagai R-TERIKAT-MP-NFRNS (R-TERIKAT-MP dengan cincin-sesuai-seterusnya yang tiada skala) digabungkan dengan analisis yang tepat, yang dinamakan analisis hyperperiod dan model tarikh akhir; kekangan keras yang lemah dan μ-corak di bawah penjadualan keutamaan statik. Kemudian, alat simulasi Matlab digunakan untuk mengesahkan hasil daripada analisis. Daripada hasil penilaian, ia membuktikan bahawa pendekatan yang dicadangkan adalah melebihi prestasi pendekatan yang sedia ada dari segi memenuhi tarikh akhir tugas-tugas.

Kata kunci: Tugas masa nyata keras yang lemah, penjadualan dibahagikan, sistem multi pemproses, analisis hyperperiod, model tarikh akhir

1.0 INTRODUCTION

Every task in the system which is completed in period of time is referred to a real-time system. For real-time systems, it should be predictable and ensure that all timing constraints will always be met. Guaranteeing temporal correctness is the endmost goal of real-time systems analysis where verifying a priori that no task has always missed the deadline, or if the deadline be missed, it missed by not more than a certain amount of time. The temporal correctness is depending on the task that has being scheduled. Due to the need to validate temporal correctness of the tasks, real-time scheduling algorithms need to be utilized and schedulability tests must be derived.

In order to determine when to execute which task on which processor if it has more than one processor, a scheduling algorithm could be used. In order to determine whether a real-time system can run within the timing constraints put upon it, a number of different algorithms have been designed to analyse a system and determine whether it is schedulable or not. The timing constrained requirements are the direct input for the scheduling analysis algorithms. But, schedulability analysis is needed to analyze the satisfaction of tasks either its deadlines can be met or not. Schedulability analysis is a mathematically sound way of predicting the timing behaviour of a set of real-time systems [1]. It is used in many different ways such as at design time or at run-time. In order to do the schedulability analysis and design for real-time systems, a task execution time and a task period must be obtain because both information are needed in related scheduling parameters especially for periodic tasks.

Timing requirements or constraints are being defined in terms of deadlines for the activities. Computations occurring in a real-time system that have timing constraints are called real-time tasks. The classification of real-time systems or tasks divided into two categories, based on the "seriousness" of deadline misses – hard real-time tasks and soft real-time tasks [2]. The consequences of deadlines miss for a hard real-time task system cannot be tolerated because some failure can affects the systems performance whereas "some" deadline misses are tolerated for soft real-time task systems. An automobile braking system is an example of a hard real-time system. When the driver pressed the brakes, the automobile can meet with an accident, if the systems are not appropriately responds. On the other hand, consider an online transaction system as a soft real-time example in which some laxity of task deadlines are tolerated; the users does not mind if during the processing of their transaction, a little delay happens but within "acceptable" limits. Nowadays, most real-time applications consist of a mix of hard and soft real-time tasks, and it called weakly hard real-time system.

A typical example of systems with weakly hard real-time requirements is multimedia systems, such as videophone application because some delay during execution is acceptable in that system and it is unnecessary to meet all the tasks deadlines as long as the misses (or deadlines) are specified precisely. In a weakly hard real-time system, the number of deadlines that may be missed can be specified. This makes a weakly hard real-time system stronger than a soft real-time system.

Such weakly hard real-time scheduling approach has traditionally focused upon scheduling analysis on uniprocessor. As a system becomes more complex and significantly increasing functionality, attention has been given to multiprocessor scheduling, comprised of several processors. Research on multiprocessor scheduling has mainly focused on ensuring strict compliance deadlines and several scheduling techniques were adopted. In an effort, Carpenter *et al*. have cataloged multiprocessor real-time scheduling algorithm considers the degree of migration jobs or tasks and the complexity of the mechanism of priority [3]. The classes of priorities includes such as (a) *static priority*, where task priorities are never change, Rate Monotonic Algorithm (RMA); (b) *dynamic priority*, where job or task priorities are change dynamically, Earliest Deadline First (EDF). The classification of multiprocessor scheduling is divided into partitioned and global [3].

2.0 RELATED WORKS

There have been some efforts that relate to schedule weakly hard real-time tasks on multiprocessor systems. Wu and Jin proposed the classical weakly hard real-

time scheduling algorithms, namely Distance Based Priority (DBP) to apply into multiprocessor applications to guarantee QoS of both hard real-time tasks and multimedia streams even under overload conditions [4]. In fact, the DBP algorithm originally was introduced by Hamdaoui and Ramanathan on uniprocessor system [5]. However, even when the system is under loaded, MPDBPs do not maximize the performance and the task deadline satisfaction ratio even when possible, which eventually fails to provide the best quality of service for multimedia streaming applications.

Another work is done by Kong and Cho where they designed a new dynamic scheduling algorithm known as the Guaranteed Multiprocessor Real-Time Scheduling (GMRTS-MK) algorithm for (*m*,*k*)-firm constrained tasks on homogenous multiprocessors [6]. However, the algorithm that they used will cause an increase in the ratio caused by the increasing number of tasks. Later, the same author, Kong and Cho introduced an Energy-constrained Multiprocessor Real-Time Scheduling (EMRTS-MK) class algorithms for weakly hard real-time for (*m*,*k*)-firm deadline constrained tasks running on multiprocessor. Multimedia service quality is maximized under energy-constrained when the static and dynamic EMRTS-MKs that they propose is using weakly hard real-time constraints [7]. However, EMRTS-MKs considers energy to make its scheduling decision.

Several static and dynamic approaches on partitioned multiprocessor scheduling have been proposed but most prior research is limits for hard and soft real-time tasks. The most well-known static priority scheduling algorithm is Rate Monotonic-First Fit (RM-FF), which use First-fit-bin-packing algorithm [8]. Thereafter, another algorithm called R-BOUND-MP-NFR (multiprocessor next-fit-ring) is presented with the best results of partitioned static priority scheduling reaching 50%. It introduces the NFR heuristic into the R-BOUND-MP algorithm [9]. Here, R-BOUND-MP is a previously known multiprocessor (MP) scheduling which combines R-BOUND [10] with First-Fit bin-packing algorithm and exploits R-BOUND. Then, Andersson in his PhD thesis proposed a new algorithm, called R-BOUND-MP-NFRNS to deal with maximizing the capacity used for the tasks without missing a deadline [11]. These stated researchers are using the allocation algorithm in their approaches.

Later, AlEnawy and Aydin adopted partitioned scheduling consider to cater the problem of energy minimization for periodic pre-emptive hard real-time tasks that are scheduled on an identical multiprocessor platform with dynamic voltage scaling capability [12].

Fisher *et al*. presented a polynomial-time partitioning approach for general sporadic task systems on an identical multiprocessor platform when static-priority scheduling policies are used on each processor [13]. This approach was adopted by Niemeier *et al* specifically deriving polynomial-time algorithms for solving several partitioning real-time scheduling problems on heterogeneous multiprocessor platforms [14].

Chishiro and Yamasaki performed experimental evaluations of partitioned semi-fixed priority scheduling in the extended imprecise computation model of multicore systems for comparable overhead [15]. Afterwards, Fan *et al*. presented a new partitioned scheduling approach to schedule fixed-priority periodic real-time tasks on multi-core platforms under Rate Monotonic Scheduling (RMS) policy [16]. Both approaches that they proposed are used for multicore systems.

There are some simple dynamic priority scheduling algorithm that using the allocation algorithms, such as EDF-FF (Earliest Deadline First-Fit), EDF-BF (Earliest Deadline Best-Fit) and Earliest Deadline Next-Fit (EDF-NF) [17]. The following two approaches were used on identical multiprocessor platform. Muller and Werner combined partitioned scheduling with EDF and used the classical approach of bin-packing with utilization bounds [18]. They considered periodic and preemptive with implicit deadlines on an identical multiprocessor. Later on, Baruah studied the partitioned EDF scheduling of sporadic task systems on identical multiprocessor platforms [19].

In most cases, if the task set is fixed and known a priori, the partitioning approach is becoming the most appropriate solution. Based on the review, in the past, partitioned scheduling of multiprocessor systems has been extensively studied but the used of scheduling analysis approaches of partitioning real-time scheduling is extremely applied for hard real-time tasks. Hence, due to this limitation, it is required to find the solution for applying partitioned scheduling approach for weakly hard real-time tasks where violation of task is allowed.

Unfortunately, the existing works on scheduling and schedulability analysis techniques of weakly hard real-time tasks are focused on global scheduling approach, thus, the aim of this paper is to contribute towards guaranteed weakly hard real-time scheduling by adopting partitioned scheduling approach and static priority assignment policy and the implement of task can be forecast. Static priority scheduling is considered than dynamic priority because it is an attractive option for real-time applications since it can ensure both the predictability of worst case behaviour and high resource utilization.

The objective of the paper is to guarantee the satisfaction of the timing constraints and parameter of weakly hard real-time systems. Solutions to the problem are by providing a schedulability analysis of periodic and preemptive tasks that are constrained by weakly hard real-time temporal constraints on multiprocessor.

3.0 A WEAKLY HARD SCHEDULING APPROACH

The main focused of research on multiprocessor scheduling is on guaranteeing the completion of deadlines. However, for weakly hard real-time tasks where violation of a task is allowed, we proposed the weakly hard scheduling approach to cater task allocation problem in partitioned scheduling. In this

approach, we adopted the technique by using bin-packing algorithm because it is known as the best heuristics that solved the task allocation problem.

Also, we used well-known, widely used and a useful performance metric, worst case utilization bounds. Within the context of preemptive scheduling of periodic tasksets, in order to make the multiprocessor scheduling algorithm resilient with exceptions, the algorithms must be combined with specific techniques such as hyperperiod analysis first to recover from deadline missing.

Due to the limitation of previous approaches wherein most of the partitioned scheduling algorithm is applied for hard real-time tasks, thus, we propose a solution to be applied for weakly hard real-time tasks (i.e. weakly hard real-time requirements, such as deadline models).

To the best of our knowledge, the weakly hard scheduling approach that we proposed is the first static real-time scheduling approach that provides a guaranteed performance for weakly hard real-time tasks on multiprocessor partitioned scheduling. We illustrate the flow of weakly hard scheduling approach of partitioned scheduling in Figure 1.

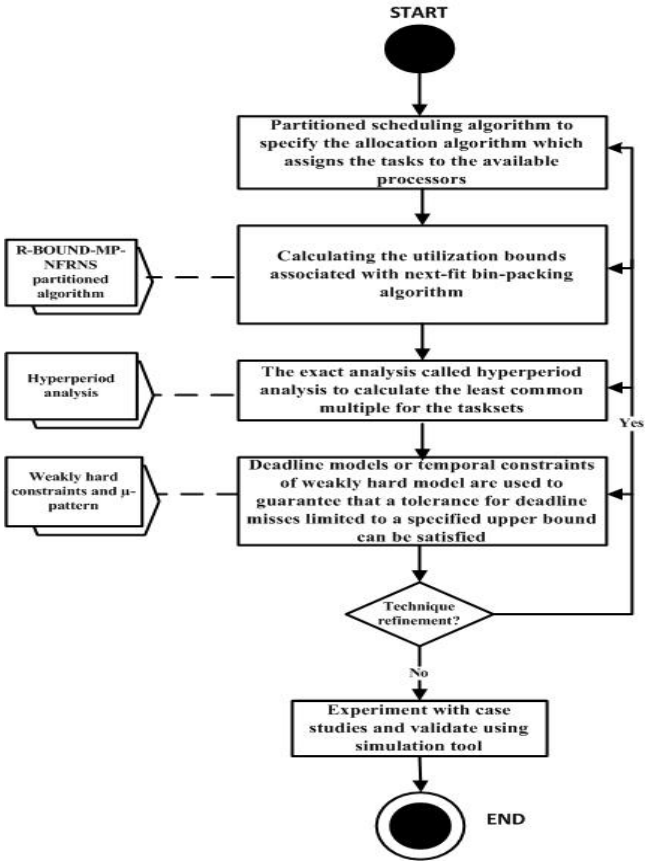


Figure 1 A weakly hard scheduling approach

4.0 THE PARTITIONED SCHEDULING

In the partitioning approach [3] (also known as offline processor assignment), no migration is allowed, where tasks are statically partitioned and allocated to processors (i.e. there is a need to choose a processor for all tasks and then run local scheduler on each processor with no migration. This is because; partitioned scheduling algorithms partitioned a task set into groups beforehand or in other words, at design time. Here, the priority is to statically assign a set of periodic tasks to a set of processors. A separate ready queue is held by each processor such that a specific processor is assigned by each task group. In other words, during the run-time, no migration of tasks is permitted from one processor to other processor. Thus, multiprocessor scheduling is equivalent to multiple uniprocessor system.

Additionally, the schedulability of partitioned scheduling can be verified by using well-understood uniprocessor analysis techniques. The main advantage of partitioning approaches is that it reduces a multiprocessor scheduling problem to a set of uniprocessor ones and partitioning approaches are widely used by system designers and received greater attention by researchers.

As shown in Figure 2, in partitioned scheduling, arrival jobs of tasks using a partitioning algorithm first is due to assign tasks to processors. Then, the local job queue is then used to place the generated jobs of each task. After that, in order to schedule jobs to each processor, a uniprocessor scheduling is used.

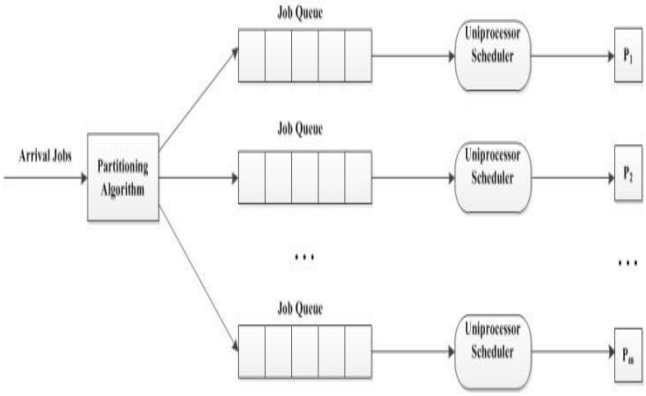


Figure 2 A multiprocessor partitioned scheduling

5.0 THE BIN-PACKING ALGORITHM

Each task having its own dedicated processor, as partition method divides tasks into partitions. Many heuristic have been proposed for partitioning. The bin-packing algorithm appears to be the popular choice [20]. In order to determine whether or not a given task can be allocated to a given processor, the bin-packing algorithm often depends on schedulability test. The following equation is to use as knowledge in schedulability test:

$$\sum_{i=1}^{n_p} \frac{C_i}{T_i} \leq B(r_p, n_p) \qquad \text{(Eq. 1)}$$

This condition uses information of the periods of the task set to achieve a high utilization. Note that in order to schedule tasks on processor p, rate monotonic algorithm is used. Thus, p is defined by the number of task assigned to processor p which denote by n_p. The fraction between the maximum and the minimum period among the tasks assigned to processor p is denoted by r_p. r denotes the set of all n tasks and it holds that $\forall_p : 1 \leq r_p < 2$. Let Equation 1:

$$B(r_p, n_p) = n_p(r_p^{\frac{1}{n_p}} - 1) + 2 / r_p - 1 \qquad \text{(Eq. 2)}$$

The algorithm R-BOUND-MP-NFRNS (R-BOUND-MP with next-fit-ring noscaling) [11] is used to partition the scheduling algorithm which is derived from the multiprocessor scheduling algorithm, R-BOUND-MP (combined R-BOUND [10] with a first-fit-bin-packing algorithm). R-BOUND is a uniprocessor scheduling algorithm for partitioned scheduling. The R-BOUND-MP-NFRNS requirements are elaborated as follows:

i. Sorting of tasks in the ascending order of periods where the first task needed to be considered is the task with the smallest period.
ii. Each uniprocessor uses Equation 1 in a schedulability test.
iii. The next-fit-bin-packing algorithm is used to assign each task.
iv. The task is assigned to processor 1 upon failure of the task assigned to processor p.

6.0 HYPERPERIOD ANALYSIS

Another alternative way to study the performance of a scheduling algorithm or to verify the schedulability of a task set, is through the use of an exact analysis such as hyperperiod analysis. Here, it can be used to verify if a task set misses any deadline, showing if it is unschedulable. Additionally, it can also be used as sufficient condition for unschedulability. The task set hyperperiod (0, H(τ)) is a feasibility interval for implicit and constrained deadline synchronous periodic tasksets, when scheduled by a deterministic and memoryless algorithm, such as RMA [21].

For these algorithms, the schedulability of the task set can be verified by checking if the schedule generated misses any deadline using the least common multiple calculation. It is possible to use hyperperiod analysis on multiprocessor scheduling because of using periodic task with deadline and preemptive fixed priority scheduling.

The higher order period or the hyperperiod, h_i consists of the number of invocations of a task in the hyperperiod at level i, $a_i = \frac{hi}{Ti}$ [22]. The least common multiple tool is used in order to get values of the hyperperiod. The hyperperiod h_i equation is given by [22]:

$$h_i = lcm\{T_j \mid \in hep(r_i)\} \qquad \text{(Eq. 3)}$$

lcm is the least common multiple of the periods of the tasks and $hep(r_i)$ is the set of tasks with a priority higher than or equal to task r_i.

7.0 WEAKLY HARD CONSTRAINTS AND μ-PATTERN

Bernat *et al.* defined the weakly hard constraints in order to precisely specify how many deadlines may be missed and met [22]. The consecutiveness of lost deadlines is very sensitive for some systems while others are only sensitive to the number of deadlines missed. The merger of the two judgments, (a) consecutiveness vs. non-consecutiveness, and (b) missed vs. met deadlines concretely guides to four basic constraints ($n \geq 1$, $n \leq m$).

i. A task r "meets any n in m deadlines", denoted $\binom{n}{m}$, if, in any window of m consecutive invocations of the task, there are at least n invocations in any order that meet the deadline.
ii. A task r "meets row n in m deadlines", denoted $\left\langle \begin{matrix} n \\ m \end{matrix} \right\rangle$, if, in any window of m consecutive invocations of the task, there are at least n consecutive invocations that meet the deadline.
iii. A task r "misses any n in m deadlines", denoted $\binom{\overline{n}}{m}$, if, in any window of m consecutive invocations of the task, no more of n deadlines are missed.
iv. A task r "misses row n in m deadlines", denoted $\left\langle \begin{matrix} \overline{n} \\ m \end{matrix} \right\rangle$, if, in any window of m consecutive invocations of the task, it is never the case that n consecutive invocations miss their deadline.

The term r denotes the set of all possible weakly hard constraints of these four types. The four constraints or cases are summarized in Table 1.

Table 1 Weakly hard constraints

	Met deadlines	Missed deadlines
Any order	$\binom{n}{m}$	$\overline{\binom{n}{m}}$
Consecutive	$\langle {n \atop m} \rangle$	$\overline{\langle n \rangle}$

For example, $\binom{4}{5}$ expresses that the task has to meet 4 deadlines in every 5 invocations. Note that this is equivalent to $\binom{1}{5}.\binom{2}{4}$ meaning that at least 2 deadlines have to be met in any 4 consecutive invocations. Note that this allows 2 deadlines to be missed consecutively.

On the contrary, $\langle {2 \atop 4} \rangle$ means that in any 4 consecutive invocations, at least 2 consecutive deadlines have to be met. This also means that only $\overline{\text{one}}$ deadline can be missed in any 4 invocations. $\langle {3 \atop 6} \rangle$ means that no more than 3 deadlines can be missed in a row in 6 consecutive invocations of the task.

A μ-pattern is a pattern of deadlines missed during a period of time. A μ-pattern is denoted by a 0 (zeros) that represent missed deadlines and by a 1 (ones) represent met deadlines. The sequence of zeros and ones can be used to characterise a task. For example, a task that has a pattern or sequence 11001101 and it does satisfy a $\binom{2}{4}$ constraint because there is at least "1's" in any 4 consecutive symbols, however it does not satisfy $\binom{1}{2}$ because there is a sequence that have 2 consecutive "0's" without a 1.

8.0 AN INERTIAL NAVIGATION SYSTEM (INS) CASE STUDY

We chose the INS case study based on the one described by Borger because it consists of a mixture hard and soft tasks [23]. Thus, it is unnecessary for the system to meet the entire task deadlines as long as the misses (or deadlines) are spaced distantly/evenly or, in other words, is weakly hard real-time tasks. We first present a task set derived from the INS system that consists of six tasks and all the tasks are listed in Table 2.

Table 2 The task set of INS case study

Task	T_i	C_i
Attitude Updater (T_1)	10	1
Velocity Updater (T_2)	15	4
Attitude Sender (T_3)	20	10
Navigation Sender (T_4)	50	20
Status Display (T_5)	50	20
Position Updater (T_6)	100	10

The six main subsystems of the INS used in the case study consist of *Attitude Updater*, *Velocity Updater*, *Attitude Sender, Navigation Sender, Status Display* and *Position Updater*. Each subsystems has deadline that equals its period. The INS system is the executive subsystems that support the scheduling of the INS task set via the real-time task dispatcher.

Among the six tasks, four tasks are known as hard tasks and another two tasks are known as soft tasks. The *Attitude Updater*, *Velocity Updater*, *Attitude Sender* and *Navigation Sender* tasks are specified as hard periodic tasks. Meanwhile, the *Status Display* and *Position Updater* tasks are specified as soft periodic tasks.

In the parameters, T_i is the period of the task, and D_i refers to the relative deadline of the task that must be finished. Every task is periodic and we assumed that $D_i = T_i$. C_i represents the worst case execution time of the tasks.

9.0 A SCHEDULABILITY ANALYSIS OF PARTITIONED SCHEDULING

A schedulability test or analysis is a condition in order to know whether a task set meets its deadlines or not. The system behaves in partitioned scheduling as explained in details as follows. The processor is assigned with each task in order to assign a local (for a processor) with static priority.

Attitude Updater has the highest priority because the task priority was decreasing in each processor. Consider six tasks in Table 3, where these tasks needed to be scheduled using 2 processors with R-BOUND-MP-NFRNS.

Table Task parameters of the task set and processors

Task	T_i	C_i	U_i	P_i
Attitude Updater (T_1)	10	1	0.1	1
Velocity Updater (T_2)	15	4	0.3	1
Attitude Sender (T_3)	20	10	0.5	2
Navigation Sender (T_4)	50	20	0.4	2
Status Display (T_5)	50	20	0.4	1
Position Updater (T_6)	100	12	0.1	1

Furthermore, the algorithm is responsible to sort the task periods in ascending order. On this analysis, processor 1 is the current one with the tasks has been assigned in order. Moreover, *Attitude Updater* has been selected and assigned into processor 1. Afterwards, *Velocity Updater* has been tried to assign into processor 1, which is successful due to the T_2/T_1 = 1.5 and n_1 = 2 gives the utilization sum for these two tasks is 0.4.

The task named *Attitude Sender* is attempted to be assigned to Processor 1 where it fails due to the max (T_1,T_2,T_3)/min (T_1,T_2,T_3) = 2.0 and n_2 = 3 the utilization sum of these three tasks is 0.9 resulting to the task being assigned to processor 2. Next, processor 2 is identified as the current processor. Afterwards, the processor 2 has been assigned with task named *Navigation Sender*. Here, task assignment is successful due to the T_4/T_3 = 2.5 and n_2 = 2 values, in turn, gives the utilization sum of the tasks to 0.9.

Later on, the task named *Status Display* is attempted to be assigned into Processor 2 where it fails due to the max (T_3,T_4,T_5)/min (T_3,T_4,T_5) = 2.5 and n_1 = 3 the utilization sum of these three tasks is 1.3. Hence, it is assigned to processor 1. Next, processor 1 is identified as the current processor. The task named *Position Updater* has been tried to assign to processor 1, which is successful due to the max (T_1,T_2,T_5,T_6)/min (T_1,T_2,T_5,T_6) = 10.0 and n_1 = 4 giving the utilization sum for these four tasks of 0.9. Table 3 shows which processor of each task is assigned.

Later, in Figure 3 we show a timing diagram which has been designed for a partitioned static priority scheduling algorithm along with the utilization bound *U*, wherein $U_i = C_i/T_i$. From the figure, it should be noted that task *Position Updater* missed its deadline by two time units. This is made task *Position Updater* unschedulable.

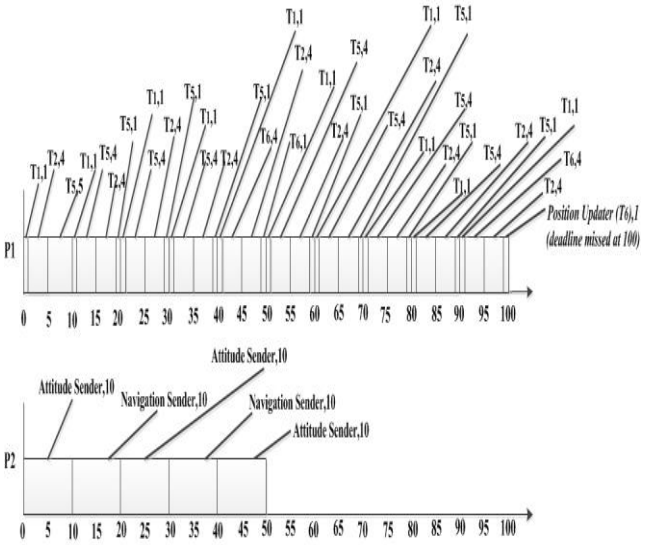


Figure 3 Timing diagram of partitioned scheduling for INS

Hence, hyperperiod analysis and weakly hard temporal constraints can be used to guarantee that both deadlines and constraints are satisfied if there have one or more tasks missed its deadline. In order to guarantee that task *Position Updater* meet its deadlines and satisfy its weakly hard temporal constraints, an exact analysis is perform to make the tasks predictable.

In order to show that exact number of deadlines that can be missed, hyperperiod analysis and weakly hard deadline models/temporal constraints are used.

Table 4 The hyperperiod analysis

Task	T_i	C_i	h_i	a_i
Attitude Updater (T_1)	10	1	10	1
Velocity Updater (T_2)	15	4	30	2
Attitude Sender (T_3)	20	10	60	3
Navigation Sender (T_4)	50	20	300	6
Status Display (T_5)	50	20	300	6
Position Updater (T_6)	100	12	300	3

From the results in Table 4, even though task *Position Updater* missed its deadline at the utilization bound, but by using hyperperiod analysis, the number of deadline missed for that task can be specified.

The utilization bound at each invocation within the hyperperiod at priority level 6 and as depicts in Figure 4, the task is invoked a_6 = 3 times within the hyperperiod at level 6.

The following Table 5 shows the exact distribution of the missed and met deadlines for invocations 1 to 3 of task *Position Updater*.

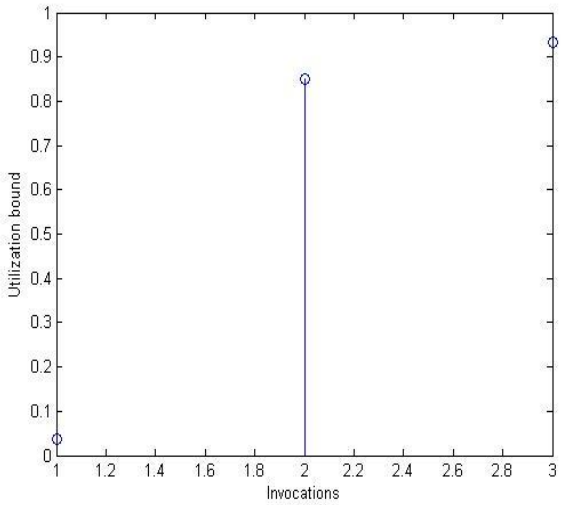


Figure 4 Invocation of *Position Updater* in the hyperperiod

Table 5 The exact distribution of the task

Task *Position Updater*	
Invocations	***μ-pattern***
1 - 3:	100

As can been seen, task *Position Updater* missed its deadline at its second and third invocations. Using weakly hard constraints, we can precisely specify the number of deadline met and missed for the task. So, task *Position Updater*'s μ-pattern would be 100. A 0 represents a deadline missed and a 1 a deadline met. It can miss it at most 2 times during its hyperperiod, (2 times every 3 invocations).

Thus, the weakly hard constraint for task *Position Updater* is defined as $\binom{1}{3}$ constraint. Checking *Position Updater*'s weakly hard constraint shows that it is satisfied, despite the miss and we can concluded that the system is weakly hard schedulable. The main objective of weakly hard constraints is to guarantee the tasks meet their timing constraints even though during execution time there are some deadlines may be missed. Meanwhile, μ-patterns are used to determine how the deadline can be missed in terms of the consecutiveness or non-consecutiveness of such missed and met deadlines.

10.0 RESULTS AND DISCUSSION

In this section, we give the videophone application case study as benchmark [24]. A set of benchmark's characteristic as given in Table 6. It consists of a set of tasks and in the second column; we listed the number of tasks. The range of these tasks execution times and periods as depicts in the third and fourth columns. The last column shows utilization. We applied the task set of videophone application into our experiment and the results are shown in Table 7.

The cases analysed by Bernat *et al*. show that soft real-time systems are not that soft as it is generally required to specify upper bounds on the number and pattern of deadlines missed during a period of time [22]. All parameters of tasks of videophone application are listed in Table 7.

Table 6 The benchmark

Applications	Task type	Num of tasks	Execution time (ms)	Period (ms)	Total Utilization
INS	Weakly hard	6	1~20	10~100	1.8
Videophone	Weakly hard	4	10~50	40~66	2.0

Table 7 Task set of videophone application

Task	T_i	C_i	U_i	P_i	h_i	a_i
VSELP speech encoding (T_1)	40	20	0.5	1	40	1
VSELP speech decoding (T_2)	40	10	0.25	1	40	1
MPEG-4 video encoding (T_3)	66	30	0.45	2	1320	20
MPEG-4 video decoding (T_4)	66	50	0.8	2	1320	20

VSELP speech encoding has the highest priority because the task priority was decreasing in each processor. Consider four tasks in Table 3, where these tasks needed to be scheduled using 2 processors with R-BOUND-MP-NFRNS.

On this analysis, processor 1 is the current one with the tasks has been assigned in order. Moreover, *VSELP speech encoding* has been selected and assigned into processor 1. Afterwards, *VSELP speech decoding* has been tried to assign into processor 1, which is successful due to the $T_2/T_1 = 1.0$ and $n_1 = 2$ gives the utilization sum for these two tasks is 0.75. The task named *MPEG-4 video encoding* is attempted to be assigned into Processor 1 where it fails due to the max (T_1,T_2,T_3)/min $(T_1,T_2,T_3) = 1.65$ and $n_2 = 3$ the utilization sum of these three tasks is 1.2 which resulting the task is assigned into processor 2. Next, processor 2 is identified as the current processor. Afterwards, the processor 2 has been assigned with task named *MPEG-4 video decoding*. The assignment is successful as $T_4/T_3 = 1.0$ and $n_2 = 2$ giving the utilization sum of the tasks of 1.25.

Referring to Figure 5, even though the timing diagram shows that task *video decoding* missed its deadline by fourteen time units, it is weakly hard schedulable. Then, Figure 6 shows that the task is invoked $a_4 = 20$ times within the hyperperiod at priority level 4.

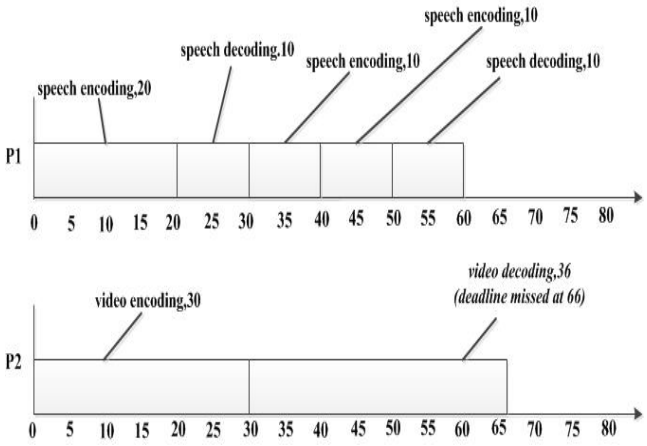


Figure 5 Timing diagram for videophone application

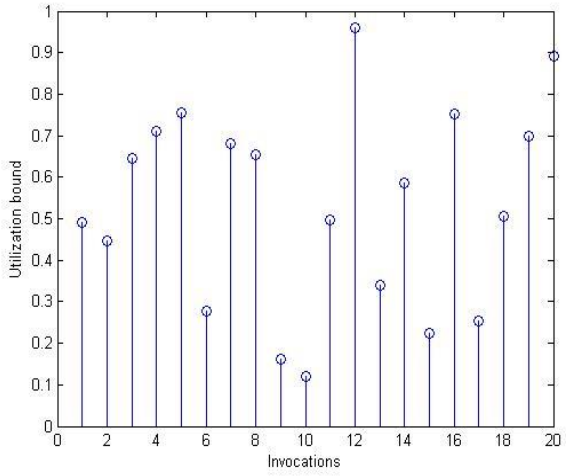


Figure 6 Invocation of *video decoding* in the hyperperiod

Table 8 The exact distribution of the task

Task *video decoding*	
Invocations	***μ-pattern***
1 - 20:	11111 11111 10111 11110

The Table 8 points the exact distribution of missed and meets deadlines for invocations 1 to 20 of task *video decoding*.

As can been seen task *video decoding* missed its deadline at its twelve and twenty invocations. Hence, task *video decoding*'s *μ*-pattern would be 11111 11111 10111 11110. Thus, the weakly hard constraint for task *video decoding* is defined as $\binom{4}{5}$ constraint.

In conclusion, after another case study, named videophone application has been used for schedulability tests; the result obtained is the number of deadlines missed for videophone application is greater than INS. This is due to the fact that the task set of videophone application is softer than INS in term of the toleration of missed deadline. Nevertheless, both tasksets are weakly hard schedulable.

11.0 PERFORMANCE EVALUATION

In this section, we evaluated the performance of our weakly hard scheduling approach in terms of the deadline satisfaction ratio. For the performance comparison between our proposed scheduling approach with one of the existing approach, EMRTS-MK was considered. We use this metric or in other words, performance measurement parameters in order to quantitatively evaluate system performance. Deadline satisfaction ratio derived from Wu and Jin [4], Kong and Cho [6, 7] and Lee *et al*. [25] defined as the number of deadlines satisfaction per total number of job releases. The deadline satisfaction ratio can be obtained by:

$$successRatio = \frac{NumberofSchedulableTasksets}{NumberofTotalTasksets} x100\% \quad \text{(Eq. 4)}$$

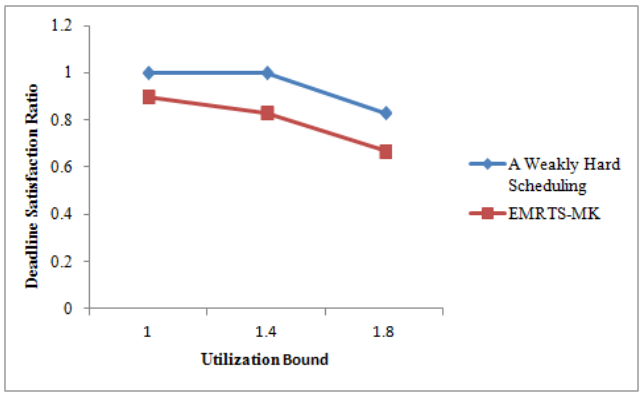


Figure 7 Deadline satisfaction ratio of tasksets

We show in Figure 7 the deadline satisfaction ratio of two different algorithms. It shows that, our proposed scheduling approach provides guaranteed performance better than EMRTS-MK.

The success ratio is an important metric which we are greatly concerned about. The success ratio defined as the number of task sets that are schedulable under a given scheduling algorithm over the total number of the task sets. A good algorithm must have a high success ratio. The Average Meet Ratio is defined as the following equation [27]:

$$AverageMeetRatio = \sum_{i=0}^{n} (Meet_i) / TotalNumberofExecution \quad \text{(Eq. 5)}$$

and the Average Miss Ratio is defined:

$$AverageMissRatio = \sum_{i=0}^{n} (Miss_i) / TotalNumberofExecution \quad \text{(Eq. 6)}$$

The number of meet of task is defined for $Meet_i$ and the number of miss of task is defined for $Miss_i$. n is the number of tasks.

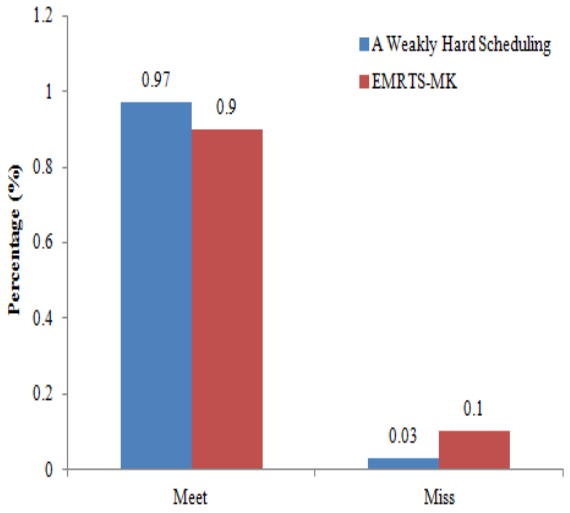


Figure 8 Meet/Miss ratio of the case study

As shown in Figure 8, the proposed scheduling approach can effectively increase the meet ratio of systems with weakly hard real-time tasks as compared to EMRTS-MK approach. This is because the number of deadline satisfactions for our proposed scheduling approach is greater than the existing ones.

The greater value of meet/success ratio demonstrates that an approach has the better performance of schedulability.

Additionally, the purpose of the comparison table in Table 9 is to evaluate the proposed scheduling analysis approach with existing approach based on the selected characteristics. Immonen and Niemelä used a framework to compare some methodologies [26], which they called Normative Information Model-based Systems Analysis and Design (NIMSAD). This framework offers an essential learning model to evaluate the crucial elements of any problem-solving scenario [27]. NIMSAD consists of four categories: context, user, content, and evaluation. Thus, the approaches or models are classified into these categories.

The first is the context category in which the classification of the method/approach is made by examining the viewpoint of the problem situation. In the user category, as the second category, the method/approach is examined from the viewpoint of the users. In the third category, the content of the method/approach itself is taken into account to examine in detail. The fourth category is the evaluation category that includes the method/approach validation. Elements in each category are defined to examine some specific requirements of the model/approach. In Table 9, the results of the evaluation are shown based on the specified criteria.

Table 9 Results of the evaluation between the existing approach and proposed approach

Category	Criterion	Kong and Cho (2012)	Proposed Approach
Context	**Scope of Applicability**	Do not propose any hybrid scheduling technique	Proposed a hybrid scheduling technique
User	**Required Extra Works**	To study one scheduling algorithm for static and dynamic schemes used	To study partitioned scheduling approach, an exact schedulability analysis and deadline models
Content	**Scheduling Approach**	Energy-constrained Multiprocessor Real-Time Scheduling (EMRTS-MK)	Hybrid schedulability analysis consists of R-BOUND-MP-NFRNS, hyperperiod analysis and deadline models (weakly hard constraints and μ-pattern)
	Schedulability Test	Pfair	Worst-case utilization bounds and bin-packing algorithm
	Performance Measurement Parameter	Dynamic failure ratio and success ratio	Deadline satisfaction ratio (meet and miss ratio)
	Predictability	To maximize the quality of multimedia services under both energy and weakly hard real-time constraint.	To guarantee the satisfaction of the timing requirements of weakly hard real-time systems.
Validation	**Maturity of the Proposed Approach**	Dynamic Voltage Scaling (DVS)	Inertial Navigation System and videophone application

12.0 CONCLUSION

In conclusion, this study has described the schedulability analysis of weakly hard real-time tasks on multiprocessor. The results analyses were investigated through an experiment with the INS and videophone application case studies. In this paper, we presented an offline static schedulability analysis in order to schedule periodic weakly hard real-time tasks. We have focused on partitioned scheduling approach.

We use R-BOUND-MP-NFRNS partitioned scheduling algorithm under the rate monotonic algorithm in order to assign each of task into processors. From the schedulability tests, task *Position Updater* missed its deadline, thus to guarantee the system is predictable, we analysed the task using hyperperiod analysis and deadline models/temporal constraints in order to know precisely the number of deadline met and missed for the task. To demonstrate the efficiency of our proposed approach, another case study is used as benchmark. Our results showed that, although the task *Position Updater* and *video decoding* missed their deadline, however, it is weakly hard schedulable This is because, the simulation result showed that the system is predictable due to the fact that, most deadlines are indeed met using exact schedulability analysis and we can obtain the number of deadlines missed and met for each task using deadline models. Most importantly, the proposed approach presented here can guarantee that both deadlines and timing constraints of the systems are successfully satisfied.

We also evaluated the performance of the proposed approach in terms of the deadline satisfaction ratio. For comparison, EMRTS-MK was considered as existing approach for deadline-constrained tasks on multiprocessor. The Matlab simulation tool is used to simulate the performance. Our experimental studies validate our results analysis and evaluation showed the effectiveness of the proposed approach and has the better performance of schedulability compared to EMRTS-MK approach in terms of achieves a higher success ratio. In fact, compared with existing approach, the proposed approach provides higher meet ratio of systems with weakly hard real-time tasks on multiprocessor.

Moreover, the results of the comparisons were derived using the selected criteria to show the differences between the approaches. It can be concluded that the proposed scheduling analysis approach can make the weakly hard real-time tasks more predictable based on used a tighter schedulability analysis and utilization-based tests performance metric.

Regarding this approach, designers can provide more predictable weakly hard real-time systems based on using weakly hard scheduling technique consists of exact analysis such as hyperperiod analysis combining with deadline models or temporal constraints, namely weakly hard constraints and μ-patterns. Also, our approach can be useful on any multiprocessor system provided that worst-case bound delays can be obtained.

Acknowledgement

The authors would like to thank profoundly to the Zamalah Scholarship from UTM, the Research Grant University (RUG) Vote No. 05H75, Ministry of Science, Technology and Innovation Malaysia (MOSTI) Vote No. 4S064, the Universiti Teknologi Malaysia (UTM) for their financial support and not forgotten, the Software Engineering Research Group (SERG) and EReTSEL lab members for their help.

References

[1] Klein. M. H. 1993. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time System.* Boston: Kluwer Academic Publisher.

[2] Bernat, G. and Burns, A. 2001. Weakly Hard Real-Time Systems. *IEEE Transactions on Computers.* 50(4): 308-321.

[3] Carpenter, J., Funk, S., Holman, P., Srinivasan, A., Anderson, J. and Baruah, S. 2004. *A Categorization of Real-Time Multiprocessor Scheduling Problems and Algorithms. Handbook on Scheduling Algorithms, Methods and Models*. Chapman Hall. CRC. 30.1-30.19.

[4] Wu, T. and Jin, S. 2008. Weakly Hard Real-Time Scheduling Algorithm for Multimedia Embedded System on Multiprocessor Platform. *1st IEEE International Conference on Ubi-Media Computing. Lanzhou.* 320-325. August.

[5] Hamdaoui, M. and Ramanathan, P. 1995. A Dynamic Priority Assignment Technique for Streams with (m,k)-firm Deadlines. *IEEE Transactions on Computers*. 44(12): 1443-1451.

[6] Kong, Y. and Cho, H. 2011. Guaranteed Scheduling for (m,k)-firm Deadlines-Constrained Real-Time Tasks on Multiprocessors. *12th International Conference on Parallel and Distributed Computing, Applications and Technologies.* 18-23.

[7] Kong, Y. and Cho, H. 2012. Energy-Constrained Scheduling for Weakly Hard Real-Time Tasks on Multiprocessors. Computer Science and Convergence. Lecture Notes in Electrical Engineering. 114: 335-347.

[8] Oh, D. I. and Baker, T. P. 1998. Utilization Bounds for N-Processor Rate Monotonic Scheduling with Stable Processor Assignment. *Real-Time Systems.* 15: 183-193.

[9] Andersson, B. and Jonsson, J. 2003. The Utilization Bounds of Partitioned and Pfair Static Priority Scheduling on Multiprocessors are 50%. In *Proceedings of the 15th Euromicro Conference on Real-Time Systems.* 33-40. July.

[10] Lauzac, S., Melhem, R. and Mosse, D. 1998. An Efficient RMS Admission Control and its Application to Multiprocessor Scheduling. In *Proc. of the IEEE Int'l Parallel Processing Symposium. Orlando: Florida*. 511-518. March.

[11] Anderson, B. 2003. Static-Priority Scheduling on Multiprocessors. PhD Thesis, Department of Computer Engineering. Chalmers University of Technology. Göteberg: Sweden.

[12] AlEnawy, T. A. and Aydin, H. 2005. Energy-Aware Task Allocation for Rate Monotonic Scheduling. *11th IEEE Real-Time and Embedded Technology and Application Symposium*. 213-223. March.

[13] Fisher, N., Baruah, S. and Baker, T. P. 2006. The Partitioned Scheduling of Sporadic Tasks According to Static Priorities. *18th IEEE Euromicro Conference on Real-Time Systems*. 127-137.

[14] Niemeier, M., Wiese, A. and Baruah, A. 2011. Partitioned Real-Time Scheduling on Heterogeneous Shared-Memory

Multiprocessors. *23rd Euromicro Conference on Real-Time Systems.* 5-8 July. 115-124.
[15] Chishiro, H. and Yamasaki, N. 2012. Experimental Evaluation of Global and Partitioned Semi-Fixed-Priority Scheduling Algorithms on Multicore Systems. *15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*. 11-13 April.
[16] Fan, M., Han, Q., Quan, G. and Ren, S. 2014. Multi-Core Partitioned Scheduling for Fixed-Priority Periodic Real-Time Tasks with Enhanced Rbound. *15th IEEE Int'l Symposium on Quality Electronic Design*. 284-289.
[17] Lopez, J. M., Garcia, M., Diaz, J. L. and Garcia, D. F. 2004. Utilization Bounds for EDF Scheduling on Real-Time Multiprocessor Systems. *Real-Time Systems.* 28(1). October.
[18] Muller, D. and Werner, M. 2011. Towards Exact Thresholds for Scheduling n Tasks on m Processors Based on Partitioned EDF. *17th IEEE Real-Time and Embedded Technology and Applications Symposium.*
[19] Baruah, S. 2011. The Partitioned EDF Scheduling of Sporadic Task Systems. *32nd IEEE Real-Time Systems Symposium*. 116-125. Vienna.
[20] Coffman, E. G., Galambos, G., Martello, S. and Vigo, D. 1998. *Bin-Packing Approximation Algorithms: Combinational Analysis.* Kluwer Academic Publishers. Ed. D. Z. Du and P. M. Pardalos.
[21] Liu, C. L. and Layland, J. W. 1993. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM.* 20(1): 46-61.
[22] Bernat, G. and Burns, A. 2001. Weakly Hard Real-Time Systems. *IEEE Transactions on Computers.* 50(4): 308-321. April.
[23] Borger, M. W. 1987. VAXELN Experimentation: Programming a Real-Time Periodic Task Dispatcher Using VAXELN Ada 1.1. Technical Report. CMU/SEI-87-TR-032 ESD-TR-87-195. November.
[24] Shin, D., Kim, J. and Lee, S. 2001. Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications. *IEEE Design and Test of Computers*. 18(2): 20-30.
[25] Lee, L-T., Tseng, C-Y. and Hsu, S-J. 2007. An Adaptive Scheduler for Embedded Multiprocessor Real-Time Systems. IEEE Region 10 Conference (TENCON'07). 1-6, Taipei.
[26] Immonen, A. and Niemelä, E. 2008. Survey of Reliability and Availability Prediction Methods from the Viewpoint of Software Architecture. *Software and Systems Modeling.* 7(1): 49-65.
[27] Kheong, L. S. and Jayaratna, N. 2009. Framework for Structuring Learning in Problem-Based Learning. Retrieved Feb 11, 2011, from http://pbl.tp.edu.sg/Understanding%20PBL/Articles/Iyejayartna.pdf.